

Comparative Study of HBase and Cassandra

Kefu Zhu

Dec. 11, 2018

Abstract

In the last decade, unprecedented growth in social media and IoT is generating huge amount of unstructured data which imposes a need for something beyond traditional relational databases. In 2009, the term NoSQL was reintroduced and became popular in the industry. The purpose of this paper is to explore and compare two major NoSQL DBMS (Database Management System) that are both wide column store type, HBase and Cassandra.

Keywords: NoSQL, HBase, Cassandra

1. Introduction

Compared to traditional relational databases, NoSQL database has apparent advantages in the era of Big Data which is characterized by high volume of data that flush in every millisecond with numerous variations in data type. First of all, NoSQL has more flexible data model which does not rely on static schema. Unlike NoSQL, relational database does not have good scalability when facing rapid data growth and is not able to handle unstructured data such as text, audio and video. Furthermore, NoSQL supports concurrent transactions which is a significant advantage over relational database due to its limitation from locking mechanism.

2. Background

2.1 HBase

HBase is an open-source, non-relational, distributed database modeled after Google's Bigtable [1] and was initially released in March 2008. It began as a project by the company Powerset, due to the need to process massive amounts of data for the purpose of natural-language search. HBase is developed as part of Apache Hadoop project and runs on top of HDFS (Hadoop Distributed File System) to provide Bigtable-like capabilities, featuring a fault-tolerant way of storing large quantities of sparse data. Although HBase does not have its own query language, technologies like Apache Hive can be put on top of Hadoop for providing SQL-like interface for data query and analysis.

2.2 Cassandra

Cassandra is also an open-source, non-relational, distributed database and was released by Facebook in July 2008 as an open-source project on Google code. It takes inspirations from both Amazon's Dynamo and Google's Bigtable. Cassandra was initially designed to fulfill the storage needs of the Inbox Search problem within Facebook. Inbox Search enables Facebook users to search through their inboxes, which was required to handle a very high write throughout, billions of writes per day, and also to scale with the number of users. Hence, Cassandra is able to handle massive amount of data across commodity servers, providing high availability with low latency operations for all clients. Cassandra is especially characterized by "treating failure as the norm rather than the exception" [2]. Unlike HBase, Cassandra has its own query language, CQL (Cassandra Query Language), which has similar syntax as SQL. CQL supports all major operation systems including Linux, Unix, OSX and Windows.

3. Architecture

HBase is based on the Master Slave Architecture Model [3], as pictured in Figure 1, which contains one master server and many region servers (salves) that lie on top of HDFS. The master server is in charge of assigning tasks and balancing workloads across region servers by leveraging Apache ZooKeeper, which is designed as a centralized service for distributed systems. Moreover, the master server is also responsible for managing and monitoring the cluster. In return, the region servers are responsible for all read/write requests for regions they serve and communicate with the clients directly.

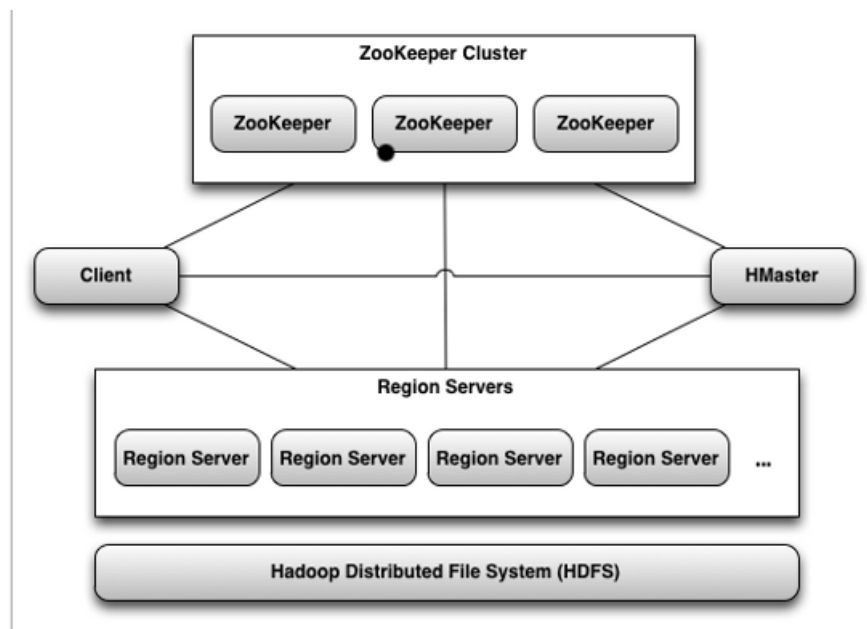



Figure 1. HBase Architecture

The data model of HBase contains row keys and column families (Figure 2). A column family is a collection of columns that share the same characteristics (example is shown in Figure 3). Each column is a collection of cells that have the same data type (column qualifier), where each cell is comprised of a value and a timestamp.

Data records within a HBase Bigtable is organized in lexicographic order by row key, where topically related data is stored close together to maximize performance.

Row ID	Column Family			Column Family			Column Family		
	Col1	Col2	Col3	Col1	Col2	Col3	Col1	Col2	Col3
1									
2									
3									

Figure 2. HBase Data Model



Row ID	Alumni Student's Personal Data		Alumni Student's Professional Data	
Alumni Student's Roll No	Name	City	Current Company	Designation
1	John	NYC	XYZ	Sr Developer
2	Yakuchi	Shanghai	ABC	Founder
3	Peter	London	GHF	CTO

Figure 3. Sample Table in HBase

Compared to the master-based structure in HBase, Cassandra is based on the Active-Active Node Model [4]. As depicted in Figure 4, all nodes are identical and are in synchronize with each other (No differentiation between master and slave). Every node contains a copy of the entire data within the cluster.

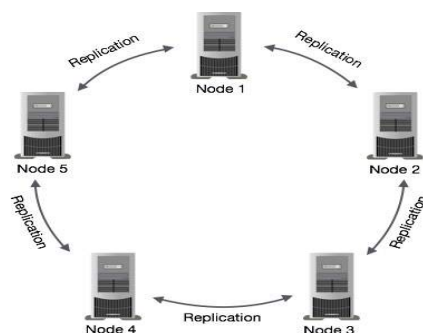


Figure 4. Cassandra Architecture

Similar to HBase, Cassandra data model is consisted with key space (similar to a database in relational database) and column families (similar to a table in relational database) [5] (Figure 5). A column family serves as a container for rows that have similar but not identical column sets. A column contains a name/value pair and a timestamp which is supplied by the client to record when it was last updated. It is not necessary to store values for all columns every time a new entity is stored.

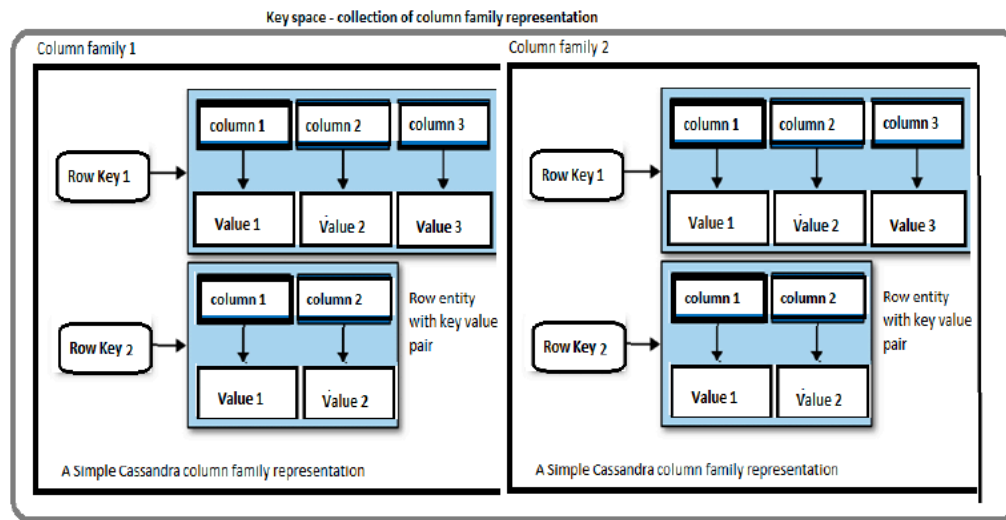


Figure 5. Cassandra Data Model

3.1 SPoF (Single Point of Failure)

Due to its master-less architecture, Cassandra is more robust than HBase in terms of single point of failure. When the only master server goes down in a HBase system, although the cluster still have some working time due to the fact that clients can communicate directly with region servers without contacting the master, the service is no longer available until the master is brought back online. When facing disruptive events, HBase is incompetent with Cassandra because Cassandra's cluster is always available. In case of any node failures, client can still read data from rest available nodes in Cassandra.

3.2 Data Consistency

Trying hard to ensure availability, Cassandra replicates and synchronizes data between nodes which leads to some data inconsistency problems. In contrast, HBase has a stronger consistent system that writes data only to one place.

3.3 DR (Disaster Recovery)

In the industry, disaster recovery is often a critical problem when facing significant disruptive events in either the infrastructure or the system. Disaster recovery is only possible in HBase if two master nodes are configured. However, since Cassandra is based off the Active-Active Node Model, every node in it contains a copy of the same data which makes recovery from disruptions quite easy.

3.4 Write Efficiency

Compared to Cassandra, HBase has faster and more consistent write operation due to the fact that it only writes on one server (Figure 6,7) [6].

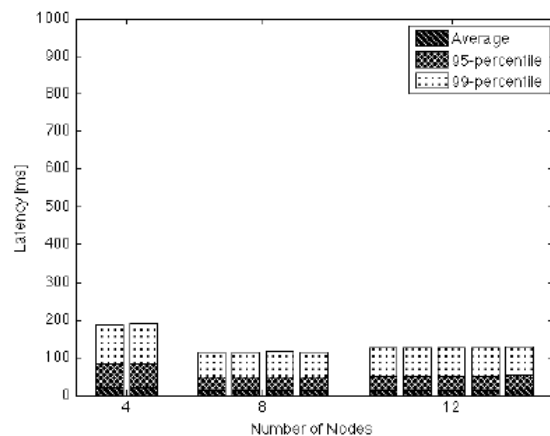


Figure 6. Cassandra Write Latency

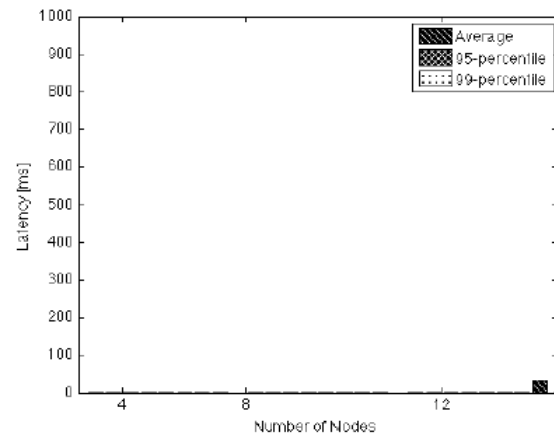


Figure 7. HBase Write Latency

4. Security

Similar to all NoSQL databases, both HBase and Cassandra have some security issues because making the effort to secure data tends to make the system heavy and drags down the performance. Nevertheless, both databases still have some features to secure data. Beyond the authentication and the authorization that exist in both HBase and Cassandra, Cassandra has client to node encryption while HBase relies on security measures within HDFS.

5. Application

As discussed in Section 3, because of the different advantages of HBase and Cassandra, Cassandra is usually more preferable if the user cannot afford any downtime from SPoF. In situations where data consistency is crucial, HBase appears to be more superior than Cassandra because it does not replicate data. In fact, Facebook migrates from Cassandra to HBase in 2010 because Cassandra's consistency model is difficult to reconcile for Facebook's new Messages infrastructure [7].

6. Conclusion

Both Cassandra and HBase are open-source NoSQL Database and can manage extremely large non-relational data sets due to high linear scalability. Despite the commons, there lies some discrepancies between Cassandra and HBase. From the design, Cassandra supports both data management and storage by itself, while HBase is only made for data management and typically relies on HDFS for data storage. The most importance difference we notice from this comparative study is that Cassandra is featured with its robustness against disruptive events, while HBase is characterized by its strong consistent system.

4. References

- [1] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber. “Bigtable: A Distributed Storage System for Structured Data”, 2006.
- [2] Avinash Lakshman, Prashant Malik. “Cassandra - A Decentralized Structured Storage System”, 2010.
- [3] “Master/slave (technology)”, Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia.
- [4] “High-availability cluster”, Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia.
- [5] V, Manoj. “Comparative Study of NoSQL Document, Column Store Databases and Evaluation of Cassandra”, 2014.
- [6] Jörn Kuhlenkamp, Markus Klems, Oliver Röss. “Benchmarking Scalability and Elasticity of Distributed Database Systems”, 2014.
- [7] Muthukkaruppan, Kannan. “The Underlying Technology of Messages”, 15 Nov. 2010, <https://www.facebook.com/notes/facebook-engineering/the-underlying-technology-of-messages/454991608919/>.